# CS152A Lab 2 Workshop 1

In this workshop, you have to get your hand dirty and write some code. There are 5 tasks in total. Some are related with the [Implementation] portion of the project, i.e. those in the ”rtl” folder, while others are related with the [Simulation] portion of the project, i.e. those in the ”tb” folder. When the you see **Demo Now**, you should demo your current results to your TA.

## [Implementation] Missing Multiply Operation

In the current version of the code, the Multiply operation described in the lab manual is missing. Take a look at how the Add operation is coded, and create the Multiply operation on your own.

Once you finish, the testbench will produce the exact same results as that of the “warm up task” from the lab manual.

## [Simulation] Nicer UART Output

The existing testbench program contains a uart model that announces each byte that’s received by the model. While generic, the output of this model is somewhat hard to read. Modify “model\_uart.v” such that the per-byte output is suppressed. Instead, model\_uart.v shall output one line at a time after a carriage-return character (‘\r’) is received. For example, print out “0003\n” instead of “0\n” “0\n” “0\n” and “3\n” for the first line of UART output.

To understand the relevant portion of code in “model\_uart.v”, you’ll need more knowledge on the UART protocol. [Wikipedia](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter) is a good start.

Note: you are only allowed to use $display in this task (e.g. no $write is allowed). We suggest that read the UART protocol to further understand the test bench before start to work on the task.

**Demo Now**. Demo your current simulation outputs to your TA. It should print out 0040, 0003, 00C0, 0100 as the value for R0, R1, R2, R4 respectively in the console outputs.

## [Implementation] Even Nicer UART Output

Currently the print instruction only prints the *value* of a register, but not the *register number*. It’s hard for us to understand what we are printing. Instead, a more intuitive way to print the content of a particular register, R0 for example, is “R0:0003”. Modify “uart\_top.v” and “nexys3.v” to make it happen.

When you finish, if you did write in your “Nicer UART Output” task, you should see the correct output format from the simulation console.

**Demo Now**. Send an arbitrary register value over the UART and demo the putty output to your TA.

## [Simulation] An Easier Way to Load Sequencer Program

The existing sequencer testbench sends a static sequence of instructions to the UUT (Unit Under Test) after the reset. Now we would like to change the static set of instructions. Instead, we will be loading instructions from a text file. The format of the file is the following:

1. The name of the file is “seq.code”
2. The file is up to 1024 lines long.
3. The first line of the file contains a binary number that indicates how many instructions are included in this file.
4. Each of the remaining (n-1) lines contains a single instruction in binary.

For example, here’s the file-equivalent of the simple sequencer instructions currently in use:

Line 1: 1001

Line 2: 00000100

Line 3: 00000000

Line 4: 00010011

Line 5: 10000110

Line 6: 01100011

Line 7: 11000000

Line 8: 11010000

Line 9: 11100000

Line 10: 11110000

**Modify the testbench such that it loads seq.code into an array, and executes every instruction in the file.**

Hint: for file I/O, you may use the built-in Verilog system task $readmemb (google Verilog quick reference), or the c-like $fopen and $fscanf tasks.

## [Simulation] Fibonacci Numbers

## Now that you have an easier way to program the sequencer from simulation, design a sequence of instructions such that the first 10 numbers of the Fibonacci series is printed from the UART.

**Demo Now**. Demo your current simulation outputs to your TA.